Student's name: ..... ID: .....

11-12

Problem 1 (2.5 points)

2.5/2.5

For each of the question below, circle either T (for **True**) or F (for **False**). **No explanations** are needed. Incorrect answers or unanswered questions are worth zero points.

(T) ~~F~~ The worst-case running time of MergeSort algorithm is $O(n \log n)$.

T (F) The worst-case running time of QuickSort algorithm is $O(n \log n)$.

(T) F QuickSort is an in-place sorting algorithm.

T (F) The worst-case running time of HeapSort algorithm is $O(n^2)$.

(T) F The worst-case running time of InsertionSort algorithm is $O(n^2)$.

Problem 2 (4 points)

4/4

(a) Give the following functions a number in order of increasing asymptotic growth rate. If two functions have the same asymptotic growth rate, give them the same number.

Function	Rank
$3 \log n + \log \log \log (n^3)$	1
$25 \lg n \quad 5 \lg n^5$	1
$\frac{n^3}{\lg n}$	2
$n^3 + 6 \log n - 5$	3
$16^{\log n} + 3 n^2$	4
4^{3n}	5

(b) Using the definition of θ, find $g(n)$, C_1, C_2, and n_0 in the following:

$$6n^4 - 3n^3 + n \log n \in \theta(g(n))$$

$6n^4 - 3n^3 + n\log n \leq 10 n^4$

$C_1 = 10$

$n_0 = 1$

$O(n^4)$

$6n^4 - 3n^3 + n\log n \geq 1 n^4$

$C_2 = 2$

$n_0 = \frac{3+1}{6-2} = \frac{4}{4} = 1$

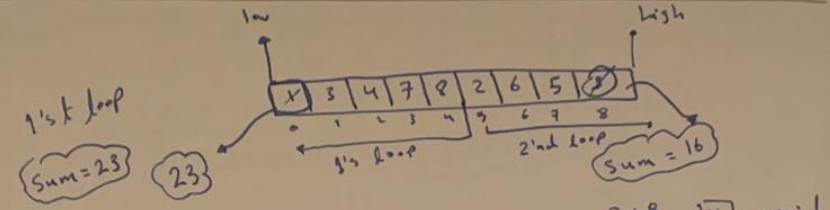
$\Omega(n^4)$

$C_1 = 10$

$C_2 = 2$

$n_0 = 1$

$\therefore \theta(n^4)$



Problem 3 (5 points) 3.5/5

Consider the pseudo-code below:

```
int maximum_sum(int A[], int low, int high)
{
    if (high == low)
        return A[low];

    int mid = (low + high) / 2;

    int sum = 0;
    for (int i = mid; i >= low; i--)
    {
        sum += A[i];
        if (sum > left_max)
            left_max = sum;
    }
```

Same

i=0; i≤ mid; i++

mid - i + 1 + 1

mid - 0 + 1 + 1

mid + 2 → (n/2) + 2

Sum + A[i]

add all nb of index

Sum stored in maxLeft

```
Algorithim ( A[], n)
for int i = 0 ⟶ n-1 { ⟶ n
   for int j = 1 ⟶ n {      n(n+1)
      if (A[i] % 2 != 0) {   n(n)
         swap ( A[i], A[j] ) n(n)
      }
   }
}
```

= 2 not sorting !!

(b) What is the time complexity of your algorithm? You need to show the step count and Big_Oh estimate.

$= O(n^2)$

$n + n^2 + n + n^2 + n^2 \rightarrow$ ✱ steps

$3n^2 + 2n \leq 3n^2 + 2n^2$

$\leq 5n^2$

$C = 5$

$n_0 = 1$

$O(n^2)$

{ i used bubble Sort algorithim }

Problem 4 (2 points) 1/9

Consider the following recurrence relation:

$$\mathbf{T(n) = 4T(\tfrac{n}{4}) + 3n.}$$

Solve this recurrence relation using recursive substitutions. Find g(n), where **T(n) = O(g(n))**.

k

[1] $T(n) = 4T(\frac{n}{4}) + 3n$

[2] $T(n) = 4[4T(\frac{n}{4^2}) + \frac{3n}{4}] + 3n$

$= 4^2 T(\frac{n}{4^2}) + \underbrace{3n + 3n}_{2}$

[3] $T(n) = 4^2[4T(\frac{n}{4^3}) + \frac{3n}{4^2}] + 3n + 3n$

$= 4^3 T(\frac{n}{4^3}) + \underbrace{3n + 3n + 3n}_{3}$

$= 4^k T(\frac{n}{4^k}) + k \cdot 3n \longrightarrow 4^{\log n} T(1) + \log n \cdot 3n$

we stop at $\frac{n}{4^k} = 1$

$n = 4^k$

$\log_4 n = k$

$= n + 3n\log n$

4

$n + 3n\log n \leq 4n$

$c = 4$

$n_0 = 1$

$O(n)$

Prove

-1

تجهيز

$T(n) = 4T(\frac{n}{4}) + 3n$

$T(\frac{n}{4}) = 4T(\frac{n}{4^2}) + \frac{3n}{4}$

$T(\frac{n}{4^2}) = 4T(\frac{n}{4^3}) + \frac{3n}{4^2}$

Problem 5 (2 points) 2/2

Solve the following recurrence using the Master theorem by giving tight θ-notation bounds. Justify your answers.

(a) $T(n) = 3T(n/3) + 3\log^3(n)$

$a = 3$ $b = 3$ $f(n) = \log^3(n)$

$\log_3^1$

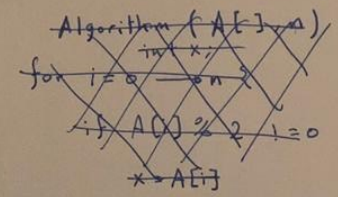
$n^{\log_3^1} = n^1 = n > \log n$

Case 1 $\Theta(n)$ ✓

2.5/4.5 Bubble insertion quicksort

Problem 6 (4.5 points)

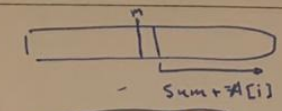
(a) Describe an in-place algorithm that takes as input an array of n integers and rearranges it so that all even integers come first and then all odd integers come next. You are allowed to use ONLY a constant amount of extra storage.



↓ next page

```
①  sum = 0;          // reset sum to 0
n  for (int i = mid + 1; i <= high; i++)
   {
       \   sum += A[i];
       \   if (sum > right_max)
               \   right_max = sum;
   }

①  int max_left_right = max(maximum_sum(A, low, mid), maximum_sum(A, mid + 1, high));
①  return max(max_left_right, left_max + right_max);
}
```



23 23 14 23 23+16 +6

a- Which problem does this algorithm solve?

returns the sum of all elements in subarray maximum

b- What is the design technique used in this solution. Explain your answer.

divide and conquere : we are spliting the array into sub arrays → divide, then adding the sum of all the sub arrays together → conquere.

it is similar to mergeSort or maybe BS

c- What is the time complexity of the following algorithm? Prove your answer.

$2T(\frac{n}{2}) + c$ n

$a = 2 \quad b = 2 \quad f(n) = 1$

$\log_2 2 = 0$

$\log_2 2$
$n = n^0 = 1 = 1$

Case 2

$\Theta(\log n)$

- 1.5